CS 348: Assignment 3 (Spring 2024)

Assigned on June 11 and due by 5:00pm EDT on June 28

Overview

For this assignment, you must use your Linux accounts and DB2 to implement an `ENROLLMENT` system consisting of two applications. The relational schema for `ENROLLMENT` must operate on the enrollment database introduced in Assignment 2 (with minor modifications) and formally defined in file `Assignment3Schema.txt` (that you need to download). Note that the file contains additional commands to recreate a bibliography database and includes sample data that correspond to the examples below. When testing your code, you should replace the sample data with appropriate alternative selections of your own test data: it is not sufficient to test your applications on the sample data provided.

`ENROLLMENT` consists of two application programs with simple command line interfaces. The requirements for the programs are given below. Finally, C together with SQL and the static embedded SQL protocol must be used to implement both application programs.

Assignment submission

Your submission to Marmoset should be a single zip file that contains your source code for the two applications and a *Makefile* that compiles the applications (you are *strongly* encouraged to use the provided `Makefile`). Marmoset will automatically prepare all appropriate tables and run tests. For each application there will be one public test and a number of secret tests. You can only see the results of the public test. You can submit multiple times. The score from your last submission will be considered. You must test your solutions locally before submitting to Marmoset to avoid clogging the build queue, which is a shared and limited resource. Also note that Marmoset builds can take a long time and last minute submissions may not return the public test results before the deadline.

Application Programs

(program `schedule`)

This application is to print a weekly schedule for the current term given a person id that can identify a professor, a student, or both. A single id will be supplied as an argument on the command line. The first line of the output should contain the person's name. For persons that are both a professor and a student it should be the professor's name (e.g. "`John Doe`"). The second line should list the current term (defined in the same way as in Assignment 2). the remaining lines should report all activities assigned to that person, namely: teaching assignments (`T`), office hours (`O`), and enrollments in classes (`C`). The format of these lines **must** follow the format below:

```
Monday     10:00:00 CS245 1 T
Thursday   15:00:00 CS245 - O*
Thursday   15:00:00 CS348 2 C*
```

Each line starts with a day of the week (10 characters), a single space, the starting time of the activity (8 characters in the `hh:mm:ss` format, a single space, the class code (5 characters), a single space, the section number (1 digit) if the activity is associated with a section and a "`-`" otherwise, a single space, and a single character indicating the type of the activity (as specified above). If a *conflict* in the schedule is detected, a `*` is added to the end of the line. A conflict in a schedule is a situation in which two or more activities are scheduled at the same time (comparing the start times is sufficient here). All such activities must be marked with a "`*`". The printed schedule must be sorted first by the activity (with teaching assignments sorted before office hours and with class enrollments last), then by day, and last by time. The schedule should be followed by a single line (terminated by `\n`) stating

```
This schedule has conflicts
```

if (and only if) conflicts are found. An example of the output of a run of the application (e.g., "`$ schedule 1`") could look as follows:

```
John Doe
S2022
Monday     10:00:00 CS245 1 T
Thursday   15:00:00 CS245 - O*
Thursday   15:00:00 CS348 2 C*
This schedule has conflicts
```

The application must print a single line (terminated by `\n`) containing "`Error`" (without any further elaboration) whenever the user asks for a non-existent person and terminate. There MUST NOT be any additional (e.g., debugging) output beyond what is stated in the specification above.

(program `enroll`)

This application's purpose is to add an activity to someone's current schedule. It is invoked with the following command line parameters (in order):

1. a person id,
2. a type of activity (i.e., `T`, `O`, or `C`),
3. day (i.e., `Monday`, ..., `Friday`),
4. time (e.g. `10:00:00`),
5. a class id (e.g, `CS348`), and
6. a section number (if needed)

The application can assume that its inputs will be well-formed (and does not have to validate these). The application should print a single line (again, terminated by `\n`) containing one of the following:

- "`Error: unknown professor/student id`";
- "`Error: assigning lecture to a non-professor`";
- "`Error: this section exists at a different time`";
- "`Error: assigning office hours to a non-professor`";
- "`Error: office hours for an unknown class`";
- "`Error: enrolling a non-student in a class`";
- "`Error: unknown class`";
- "`Warning: time conflict`" (the request conflicts with an existing one);
- "`OK`" otherwise,

without any further elaboration. The last two cases should modify the database while the "`Error:`" cases need to correctly rollback any changes. In the case of multiple errors, the application should print the top-most in the above list that applies. There MUST NOT be any additional (e.g., debugging) output beyond what is stated in the specifications and shown in the examples.

For example

```
$ enroll 1 T Friday '12.00.00' CS234 2
OK
```

succeeds assigning teaching section 2 of CS234 to professor 1 on `Friday` at noon (or create a new section, if one does not exist). Consequently,

```
$ enroll 1 O Friday '12.00.00' CS345
Warning: time conflict
```

succeeds assigning office hours for CS345 to professor 1 again on `Friday` at noon, but warns about the ensuing conflict.

```
$ enroll 9 C Friday '12.00.00' CS234 2
Error: unknown professor/student id
```

fails enrolling a non-existent person into CS234 (were 9 a professor only, the message should have been "`Error: enrolling non-student in a class`").